# Chapter 7.9

1.

## Step 1

The writes will generate a read from memory of the L2 cache line and they the line is written back to memory. The data updated in the block is updated in L1 and L2, we assuming L1 is updated on a write miss. The status of the line is set to "dirty". Specific to the coherency protocol assumed, on the first read from another node, a cache-to-cache transfer takes place of the entire dirty cache line

## step 2

Depending on the cache coherency protocol used, the status of the line will be changed the other two reads can be serviced from any of the caches on the two nodes with the updated data The accesses for the other three writes are handled exactly the same way. The key concept is that all nodes are interrogated on all reads to maintain coherency and all must respond to service the read miss

(2) Chapter 7 9

For a directory based coherency mechanism the address space of memory is partitioned on a node-by-node basis, only the directory respons for the address requested needs to be interrogated The directory controller will they initiate th cache-to-cache transfer, but will not need to bother the L2 caches on the nodes where the lin is not present. All state updates are handled locally at the directory. For the last two req again the single directory is interrogated and directory controller initiates the cache-to-cach transfer. But only the two nodes participatin in the transfer are involved. This increases the L2 bandwidth.. Since only the minimum number of cache accesses or interrogations is involved in the transaction

(3) of

Chapter 7.9

For the cache-based block status case, all coherency traffic is managed at the L2 level between CPUs. So, this scenario should not change except that reads by the 3 local cores should not generate any coherence messages outside of the CPU. For the directory case, all accesses need to interrogate the directory and the directory controller will initiate cache-to-cache transfers. Again, the number of accesses is greatly reduced using the directory approach

(A)

# Chapter 7.9

Assuming an invalidate on write policy, for writes on the same CPU, the L1 dirty copy occur on the third and fourth write. When writes are done on another CPU, they coherance management moves to the L2 and the L2 copy on the first CPU is invalidated

The local write activity is the same as for the first CPU. This repeats for the last two CPUs.

Of course, this assumes that the order of the writes is in numerical order, with the group of 4 writes being performed on the same CPU on each core. If we instead assume that consecutive writes are performed by different CPUs each time, and they invalidates will take place at the L2 cache level on each write.